05 | 案例：测试框架如何才能支持RESTful风格的接口?

2020-02-10 陈磊

接口测试实战课 进入课程 >

讲述：陈磊
时长 11:16　大小 9.04M

你好，我是陈磊。

在前面的课程中，我们一起学习了如何把流程化的测试脚本，一步一步抽象成你自己的测试框架。无论你用的是什么编程语言，封装和抽象自己的测试框架都会让你的接口测试任务事半功倍。

我相信你在平时生活或工作中，一定会接触到各式各样的软件系统，而现在的软件系统和 5 年前相比，最大差别就在于结构不同。

在我读大学的时候，绝大部分系统还都是用一个 Tomcat 来搞定的；但现在的系统更加复杂，它们已经无法只用一个 Web 中间件独立对外提供服务，它们之间都也是通过相互调用

来完成业务逻辑的，这里面既包含了服务端和服务端的调用，也包含了前端和服务端的调用，这就催生了 RESTful 风格的 HTTP 接口。

所以，这节课我就来和你讲讲，如何让你的测试框架完美支持 RESTful 风格的接口测试。这里我希望你能不断强化封装测试框架的三个流程，不断为自己的接口测试框架添砖加瓦。

不过，我不会将 RESTful 的规则一条一条念给你听，我想让你知道的重点是作为测试工程师，你要学会从测试工程师的角度观察 RESTful 接口，要学会怎么分析和验证这类接口，这也是今天我们今天这节课的主要内容。

RESTful 风格接口关我什么事?

看到这里，你是不是一脸困惑：**RESTful 是一个接口的封装风格，和我们测试人员有什么关系呢?**

要想理解它和我们测试工程师的关系，你就要先知道 RESTful 风格的接口到底有什么好。

如果你用螺丝、钉子和板材等一系列原材料组装过家具，那么你肯定见到过各种千奇百怪的螺丝，比如一字的、十字的、三角形的、六角形的和海星形的等等，为了加固这些各式各样的螺丝，你就要准备各式各样的螺丝刀，因此，你的工具箱就会被不同规格和大小的螺丝刀填满。

不知道你是不是也和我一样，面对塞满螺丝刀的、乱七八糟的工具箱，心里非常急躁。但后来我在宜家看到一款螺丝刀，它只有一个刀柄，但给你提供了一整套各种形状、各种大小的螺丝刀刀头。

这样你在使用时，只要根据螺丝规格的不同，选择替换同形状的刀头就可以了；与此同时，它们放在工具箱里面又会显得很整齐，而不会七零八落。而且，如果你后续需要使用其它特殊形状的螺丝刀，你只要买和刀柄连接口一样的刀头就可以了，而不用再买一个完整的螺丝刀了。

如果你理解了上面这个场景，也就能很好地理解 RESTful 风格的接口了。**它主要就是一组设计原则和约束条件，本质上就是让消费者依据 URI 就可以找到资源，并通过简单的服务输入输出完成服务的交互。**

它所约束的每一个 URI，都是独一无二的一个资源，通过 HTTP 的方法进行资源操作，实现表现层的状态转化。这就和螺丝刀刀头一样，待解决的问题就像螺丝，每一个接口只面向一种特定的资源，而不需要关心其他接口的处理方式，这样，你就能够一目了然地知道，该用哪种螺丝刀头拧哪种螺丝了，这就降低了接口开发的复杂度。

软件开发人员只要遵循 RESTful 的实践标准，按照一定的内部定义开发外接口，就会形成类似螺丝刀刀头一样轻便的接口，对外提供服务。现在的很多项目，无论是服务端和服务端的调用，还是前端和服务端的调用，都采用了这一种方式来设计接口。

对于我们测试工程师来说，RESTful 风格的接口还是之前的访问模式，它同样是一种 HTTP 协议的接口，同样可以使用我们上节课一起封装的框架完成接口测试的任务。

但是，它和我们之前讲过的 HTTP 协议的接口测试还是有一些区别，这些区别导致了你现在的框架还需要做一些修改，这样才能让它支持 RESTful 风格的接口测试。

让你的框架可以测试一个 RESTful 风格接口

现在，你知道 RESTful 接口和你的接口测试有很大关系了，那么，RESTful 接口的测试和原始的 HTTP 协议接口的测试，又有什么区别呢？

这里面有两部分需要你特别关注：**数据交换的承载方式和操作方式**。

我先说说数据交换的承载方式，RESTful 风格的接口主要是以 JSON 格式来进行数据交换，如果你还记得我之前提过的“Battle（战场）”这个系统，那么你一定在它的 Readme.md 中，看到了 Request 和 Response 对数据部分的一些定义，那就是 JSON。虽然“战场”不能算是一个严格的 RESTful 接口的系统，但是，在数据交接的承载方式上，它模仿了 RESTful 的样子。

另外一个部分是操作方式，在“战场”系统中，我们用了 HTTP 协议的 Get 和 Post，其实 HTTP 协议有很多方法，但是我们仅仅用了这两种，而 RESTful 的规定，使 HTTP 的很多方法都被利用到了，比如说，Get 方法用来获取资源，Post 方法用来新建资源（或者更新资源）；再比如说，Put 方法用来更新资源、Delete 方法用来删除资源等等。

在明白 RESTful 风格的接口和普通的 HTTP 协议接口的区别后，我们现在来想一想，你自己的框架需要添加什么内容，才能支持 RESTful 风格的接口呢？**这里我总结了两个方法：**

借助外力和自己封装。

借助外力

这里，我们 RESTful 的第一个数据交换的承载方式是 JSON，我们的框架在之前的“战场”系统中就已经使用了它，虽然全部的操作都是参数拼凑的过程，但这已经满足了我们的需求。

这时如果你仍要拼凑很多复杂的数据，就需要使用 JSON 字符串和代码对象实体的转换，它有一个专业的叫法：**序列化和反序列化**。这个词语听着就很难理解，所以现在，我用一个生活中的小例子来告诉你，这个晦涩难懂的概念到底是什么意思。

如果你在商场看中了一款衣柜，但它很大，为了方便运输，就必须要先把它拆掉，运到家后再重新组装。你和商家协商好了，由他们为你把这个衣柜拆成可重组的零件运到家里，然后由你自己把这些零件重新组装成一个衣柜。

那么在这里，商家把衣柜拆成各个零件、然后打包的这个过程就是“序列化”，在代码中，就是将一些程序对象转换成 JSON 等格式的字符串的过程。接下来，你用这些零件再重新组装成一个衣柜，这个过程就是“反序列化”，在代码中，就是 JSON 等格式的字符串转换成程序的对象的过程。

为了能让你的框架可以快速完成序列化和反序列化，我建议你在代码中引入一个外部支持的库，就像 Python 有 JSON 库、Java 有 Fastjson 库。这些公开的库其实都不需要做任何的修改，就可以拿来使用，所以，无论你使用哪种技术栈，这样的基础库都是存在的，你只需要在网上找一下，然后花几分钟看一下怎么使用，就可以拿到自己的框架里使用了。

自己封装

现在，我们已经可以借助开源库，解决数据交换的事情了，但是，RESTful 风格接口和普通 HTTP 接口相比，还有一个明显的区别，那就是 RESTful 规定了 HTTP 的每一个方法都做固定的事情，可我们原来框架中的 Common 类却只支持 Get 和 Post 方法，因此，你需要在 Common 类中加入 Delete 和 Put 方法的支持。具体的操作你可以依据下面这个代码段来完成：

复制代码

```
def put(self,uri,params=None):
```

```
    '''
    封装你自己的put方法，uri是访问路由，params是put请求需要传递的参数，如果没有参数这里为空
    :param uri: 访问路由
    :param params: 传递参数，string类型，默认为None
    :return: 此次访问的response
    '''
    url = self.url_root+uri
    if params is not None:
      # 如果有参数，那么通过put方式访问对应的url，并将参数赋值给requests.put默认参数data
      # 返回request的Response结果，类型为requests的Response类型
      res = requests.put(url, data=params)
   else:

      # 如果无参数，访问方式如下

      # 返回request的Response结果，类型为requests的Response类型

      res = requests.put(url)

   return res


def delete(self,uri,params=None):
  '''
  封装你自己的delete方法，uri是访问路由，params是delete请求需要传递的参数，如果没有参数这里为空
  :param uri: 访问路由
  :param params: 传递参数，string类型，默认为None
  :return: 此次访问的response
  '''
  url = self.url_root + uri
  if params is not None:
    # 如果有参数，那么通过put方式访问对应的url，并将参数赋值给requests.put默认参数data
    # 返回request的Response结果，类型为requests的Response类型
    res = requests.delete(url, data=params)
  else:
    # 如果无参数，访问方式如下
    # 返回request的Response结果，类型为requests的Response类型
    res = requests.put(url)
  return res
```

在上面的代码中，你可以看到，我们为了实现 HTTP 协议的 Put 和 Delete 方法，自己封装了 put() 函数和 delete() 函数。其实，要实现 RESTful 风格的接口测试，你只要封装 HTTP 协议对应的 Method 方法就可以了，这样，你的框架就能完美的支持 RESTful 风格的接口了。完成了这个操作后，我们的 Common 类就既可以完成 HTTP 协议接口的测试，也可以完成 RESTful 接口的测试了。

总结

到这里，我们已经结束了今天的课程了。我们今天主要完成了 RESTful 风格接口的测试，对比之前的例子以及你自己的测试框架，针对框架中 RESTful 里缺失的部分，我为你提供了对应的解决方法。

在文中我讲了很多内容，但是完成 RESTful 风格接口测试，主要是通过两步操作，来为你的测试框架添加对应接口的测试能力的：

1. 借助外力。目前网上已经有很多成熟的、各式各样的支持库，你要尽量拿来为己所用，而不要从零建设，这样，既弥补了我们开发能力不强的短板，也能提高我们的研发效率。
2. 自己封装。你要注意的是，自己封装和借助外力并不互相冲突，你要借助外力，然后将它封装到你自己的框架中，这是一个借力打力的好方法。

随着我们的课程的不断深入以及内容的不断丰富，我相信，你最终会获得一个完全适合你自己，又可以解决实际工作任务的测试框架，这也是你自己的接口测试武器仓库，里面有解决各种接口测试问题的方法。它会是一个私有仓库，里面每一个武器都是为你自己量身定制的，因此，每一件武器你用起来都会更得心应手。

思考题

我今天讲了 RESTful 接口测试，并为你的私有测试框架添加了各式各样的新武器，那么，你能用你现在的新武器，解决一个你负责的 RESTful 的接口测试吗？在今天的框架中，随着你实际工作的使用，你又有了什么样的新设计呢？

我是陈磊，欢迎你在留言区留言分享你的观点，如果这篇文章让你有新的启发，也欢迎你把文章分享给你的朋友，我们一起探讨和学习。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　04 | 案例：如何把流程化的测试脚本抽象为测试框架？

精选留言 (3)

写留言

Zh

张胜坡 2-10

可以写一个测试的实例吗？参数是data或者json 格式，如何在测试框架中怎么兼容？

作者回复: 代码示例中有使用例子，框架中引入json即可完成。谢谢留言。

roychris
2020-02-10

序列化和反序列化的具体例子后面有例子讲解吗？

展开

作者回复: 具体代码中有对应写法，但是不会重点讲解，这部分就是引出概念，对应部分在网络上很多教程，就没有在这里重复出现了。

7F
2020-02-10

这里的风格是指入参全部定义成json格式？然后不需要在代码中在去写入参了么？

作者回复: 参数还是写的只是写的方式一些变化，HTTP的Method定义了操作方法而不在随开发习惯乱用